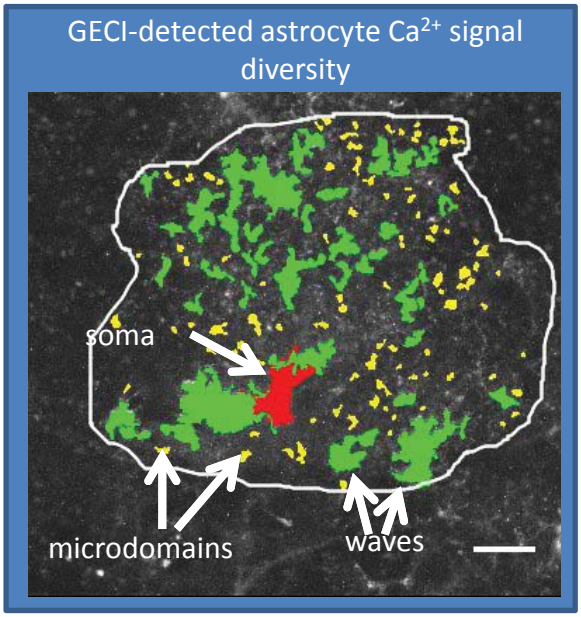
GECIquant

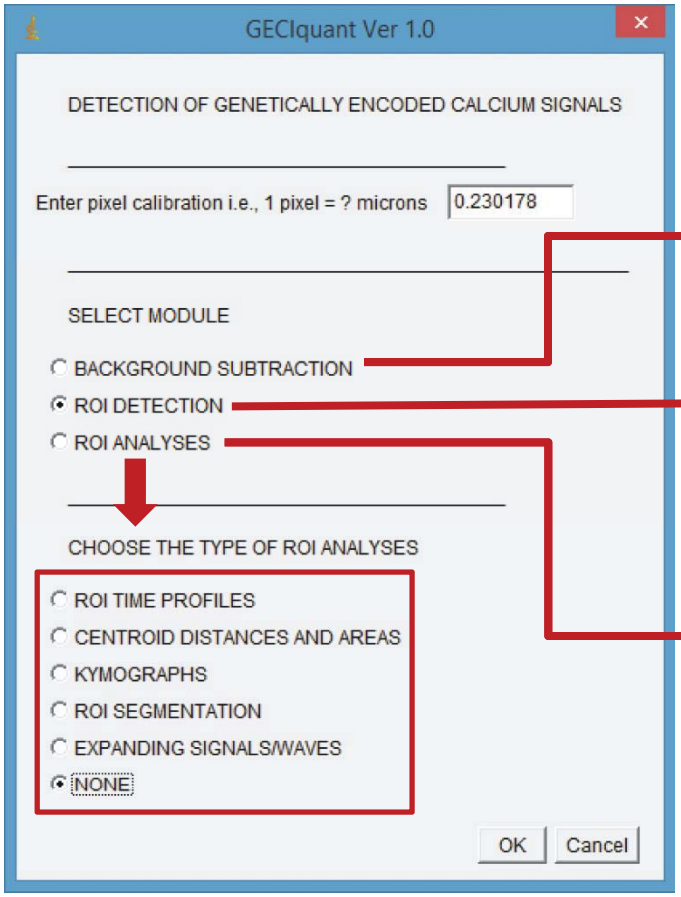
Semi-automated detection and quantification of Ca^{2+} signals measured with GCaMP6



GECIquant is a customized ImageJ script developed to detect and analyze the spatiotemporal features of Ca^{2+} signals in astrocytes and neurons acquired with fluorescence microscopy using genetically encoded calcium indicators (GECI).

Regions of interest (ROI) are detected using a sequence of image processing steps involving binary thresholding and object tracing ImageJ functions in a user interactive manner.

- ❑ To run GECIquant, copy the script file into \ImageJ\plugins folder. Run GECIquant plugin from "Plugins" menu.
- ❑ Requires ImageJ version ≥ 1.48s.
- ❑ Requires 8 or 16 or 32-bit confocal image time stack to be open.



Module-1: Preprocessing image stack to subtract background intensities in a user-interactive manner.

Module-2: Semi-automatic ROI detection in a user-interactive manner.

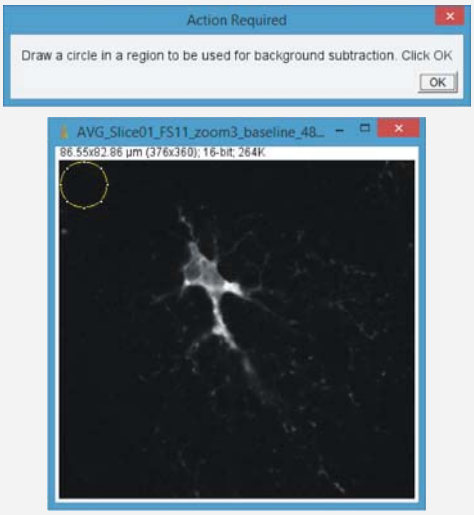
Module-3: Spatio-temporal ROI analyses by generation of time profiles, centroid distances and areas, kymographs, segmentation and expanding signal profiles.

Email: bkhakh@mednet.ucla.edu; svenugopal10@gmail.com

Developed by: **S Venugopal, R Srinivasan and BS Khakh @ UCLA**

Preprocessing and ROI detection

1 Background subtraction



1. Open image t-stack.
2. Run GECIquant; select 'Background Subtraction' radio button.
3. In the 2-dimensional average projection of the t-stack (image above), draw a circular area representing background intensities.
4. Click OK in the message window (above).

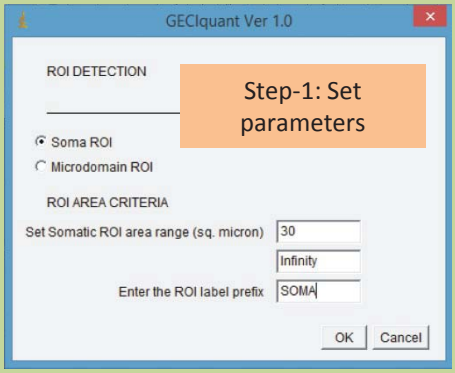
Output: Background subtracted image t-stack.

2a ROI detection

1. Open image t-stack.
2. Run GECIquant; select ''ROI Detection' radio button.
3. Click OK.
4. In the user interface that pops up (see below), follow the user-interactive steps to set thresholds for different types of ROI detection.

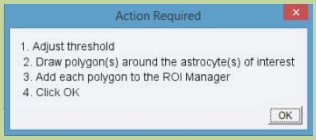
2a

Soma detection user interface



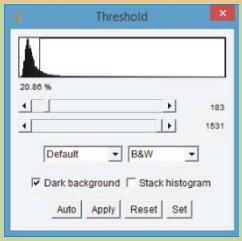
Step-1: Set parameters

Message window with instructions

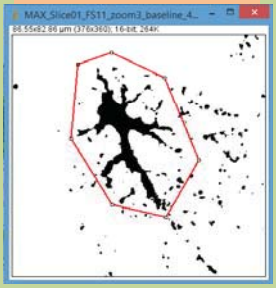


Step-5: Click OK

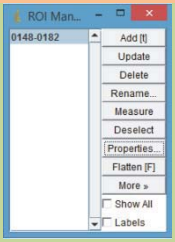
Step-2: Adjust threshold



Step-3: Draw polygon

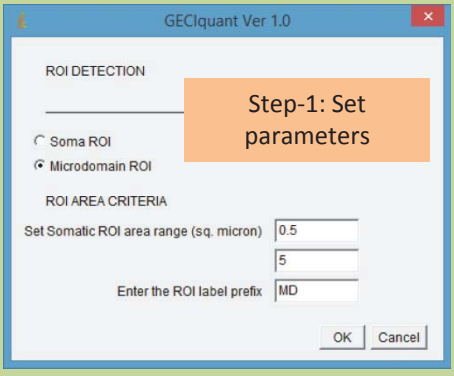


Step-4: Add to ROI manager



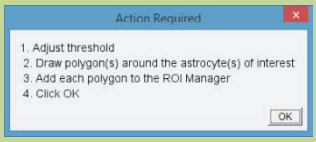
2b

Microdomain detection user interface



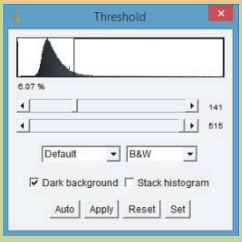
Step-1: Set parameters

Message window with instructions

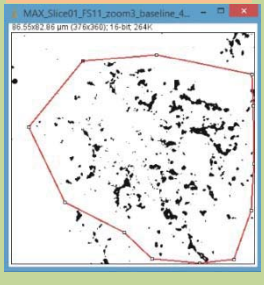


Step-5: Click OK

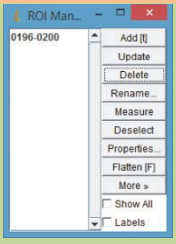
Step-2: Adjust threshold



Step-3: Draw polygon



Step-4: Add to ROI manager

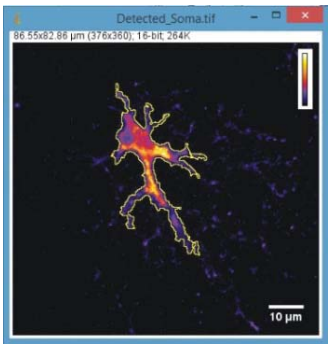


ROI outputs

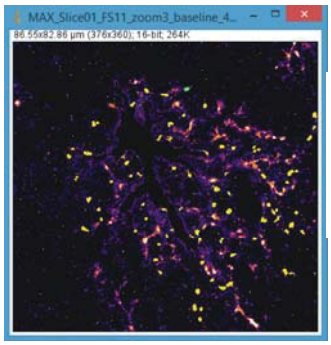
Outputs of ROI detection

1

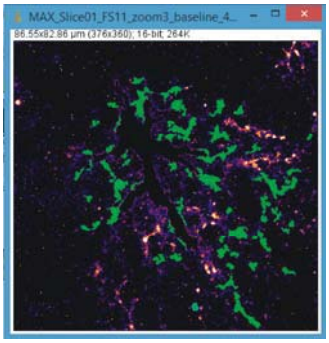
Somatic ROI



Microdomain ROI

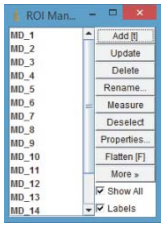


Expanding Signal ROI



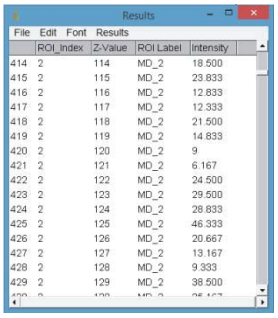
2

ROI list in ROI manager



3

ROI time profile



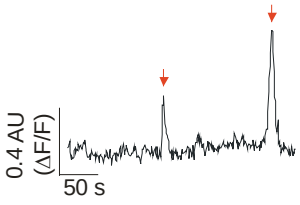
4

Thresholds

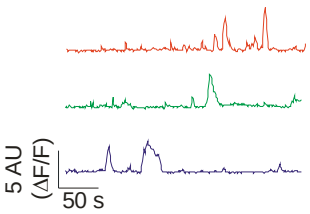
The automatic and user re-adjusted thresholds are printed into the log file. The user can save these threshold values.

Example plots of ROI profiles

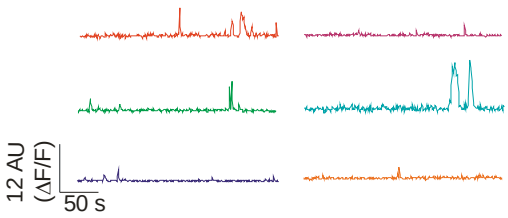
Somatic ROI Ca^{2+} transients



Expanding ROI Ca^{2+} transients



Microdomain ROI Ca^{2+} transients



These plots are drawn using a different software (e.g., OriginPro) with the time profile data output using GECIquant in ImageJ

ROI Analyses

3A ROI Time Profiles

1. Open image t-stack.
2. Start ROI Manager and open the ROI set file with the ROI list.
3. Run GECIquant; select 'ROI Analyses' radio button; then select 'ROI time profile' radio button.
4. Click OK.

Output: Results window consisting of the time or z-profile of all the ROI in the ROI manager.

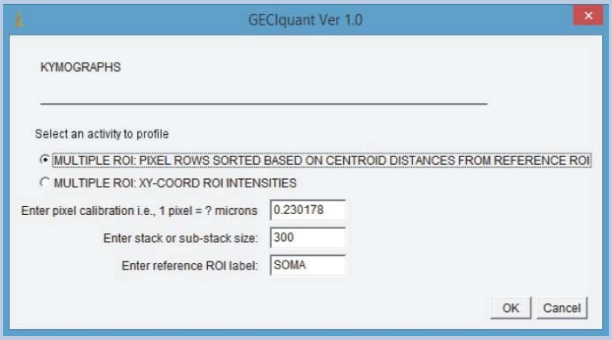
3B Centroid distances and ROI areas

1. Open image t-stack.
2. Start ROI Manager and open the ROI set file with the ROI list.
3. Run GECIquant; select 'ROI Analyses' radio button; then select 'Centroid distances and areas' radio button.
4. Click OK.
5. In the user interface that pops-up, enter the reference ROI label.

Output: Results window consisting of the centroid distances from reference and the areas of all the ROI in the ROI manager.

3C Kymographs

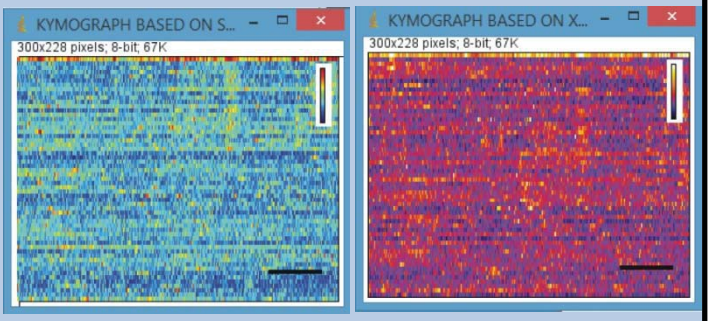
1. Open image t-stack.
2. Start ROI Manager and open the ROI set file with the ROI list.
3. Run GECIquant; select 'ROI Analyses' radio button; then select 'Kymographs' radio button.
4. Click OK.
5. In the user interface that pops-up (see below), select the type of graph and enter the parameters.



6. Click OK.

Kymograph outputs

Kymograph-1 Kymograph-2



Each row represents an ROI; x-axis shows time and y-axis shows ROI sorted based on the distance between the ROI centroid and soma centroid (left) and with ROI sorted based on x-y pixel coordinates (right)

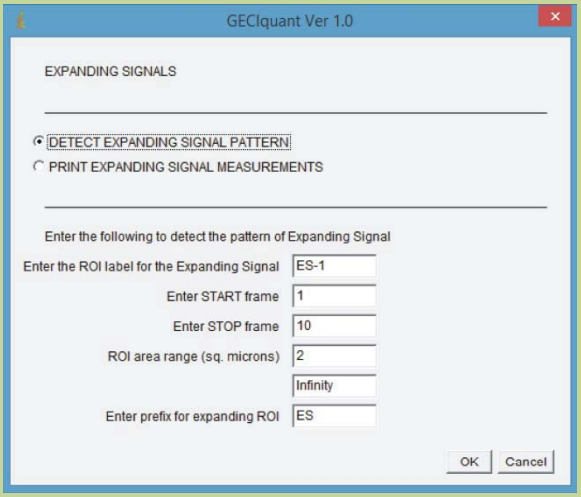
3D Expanding signals

Detection:

1. Open image t-stack.
2. Start ROI Manager and open the ROI set file with the expanding ROI.
3. Run GECIquant; select 'ROI Analyses' radio button; then select 'Expanding Signals' radio button.
4. Click OK.
5. In the user interface that pops-up (see right), select the "Detect …" and enter the parameters.
6. Click OK.

Print …Measurement:

1. With the image t-stack and ROI manager open, now open the ROI set file that consists the list of all the expanding sub-ROI previously detected and saved.
2. Click OK.

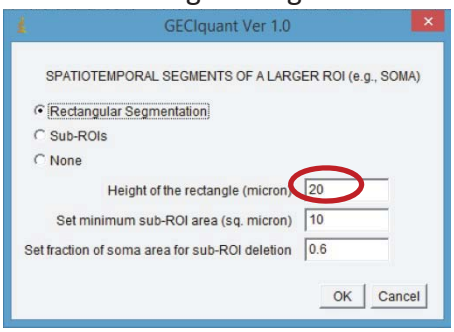


Output: i) A sub-stack consisting of the expanding profile of the ROI, ii) For detection, ROI manager lists of all sub-ROI detected, iii) For printing, sub-ROI characteristics are printed into the Results window.

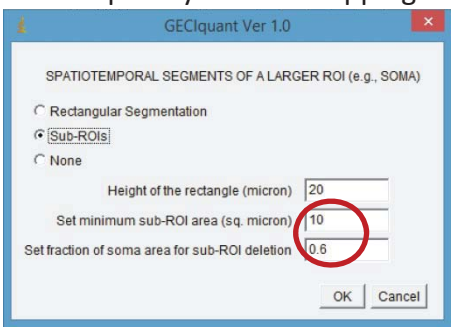
3E

Spatiotemporal segments of ROI

1. Rectangular segments



2. Spatiotemporally non-overlapping sub-ROI



1. Open image t-stack.
2. Start ROI Manager and open the ROI set file with the ROI to be segmented (e.g., somatic ROI).
3. Run GECIquant; select 'ROI Analyses' radio button; then select 'ROI Segmentation' radio button.
4. Click OK.

Rectangular Segments:

1. In the user interface that pops-up (see top left), select 'Rectangular Segmentation' and enter the height of the rectangle.
2. Click OK.

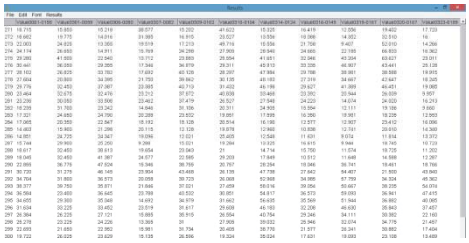
Sub-ROI Segments:

1. In the user interface (top right), select 'Sub-ROI' and set the minimum sub-ROI area to be included and the percent overlap of the largest detected sub-ROI with the somatic ROI.
2. Click OK.

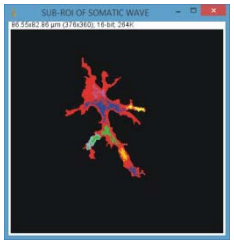
Outputs

1. Rectangular Segmentation ROI

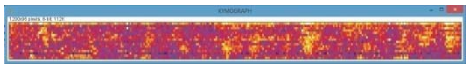
ROI time profile

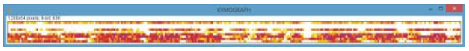


2. Sub-ROI



ROI intensity kymographs





Rectangular segmentation of somatic ROI

This approach creates smaller segments of a large ROI (e.g., soma) by placing a moving rectangle whose height is specified by the user. First, the somatic ROI is bounded by the smallest rectangle. Keeping the width of this rectangle constant, smaller rectangles of user specified height are created starting from the top left corner of the somatic rectangle. Within each smaller rectangle, the program detects regions of the signal following the same steps described for ROI detection.

Spatiotemporal segmentation of the somatic ROI

This feature extracts spatiotemporally distinct sub-ROI of a larger ROI (e.g., soma). First, every 3 frames of the stack are averaged and the resulting reduced stack (sub-stack) is used for subsequent steps. Next, in each frame of this sub-stack, the somatic ROI is detected and its area (A_{sub}) is compared to the pre-determined somatic ROI (A_{soma}) that encompasses the entire soma. If A_{sub} < X% of A_{soma}, the corresponding ROI is suggested to be a sub-ROI of the soma. Once such sub-ROI are detected in each frame of the sub-stack, the program eliminates all the spatially overlapping sub-ROI by comparing their pixel coordinates. This results in a set of sub-ROI of the soma detected based on the temporal characteristics of the somatic wave. Note that these sub-ROI are spatially distinct as well.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// PROGRAM:          GECIquant Ver 1.0                                                                                          //
// DEVELOPED BY:     Sharmila Venugopal, Rahul Srinivasan & Baljit S Khakh                                                      //
//                   DEPARTMENT OF PHYSIOLOGY, UCLA                                                                             //
// CONTACT:          svenugopal10@gmail.com (for technical questions regarding the program)                                     //
//                   bkhakh@mednet.ucla.edu (for questions regarding GECIs etc)                                                 //
// DATE:             June 2014                                                                                                  //
// DESCRIPTION:      We have developed a customized IMAGE J MACRO script that can extract the spatiotemporal                   //
//                   features of genetically encoded fluorescent Ca2+ signals in astrocytes and neurons acquired as             //
//                   image stacks using confocal microscopy. The signal consists of Ca2+ transients in the soma                 //
//                   and processes.  Based on where the Ca2+ transients are observed, the program can detect                    //
//                   diverse Ca2+ signals in different morphological compartments of a cell:                                    //
// 1. Somatic ROI encompassing the soma and the adjoining branches.                                                             //
// 2. Peripheral ROI in the branches and microdomains in the peripheral processes distant from the soma.                        //
// The program includes 3 modules: 1) Background Subtraction, 2) ROI detection and 3) ROI Analyses.                             //
//                                                                                                                              //
// If you benefit from this program, please cite our paper_____                                                                 //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Global variables
var MainStack, StackTitle;
var width, height, channels, slices, frames;
var pixel_cal;

// GECIquant Macro/Plugin
                    requires("1.48s");
                    // Clearing the Results window and Analyze menu – Set Measurements selections
                    run("Clear Results");
                    run("Set Measurements...", "redirect=None decimal=3");

                    MainStack = getTitle();
                    Stack.getDimensions(width, height, channels, slices, frames);
                    StackTitle = MainStack;

                    // Opening dialog – main user interface
                    Dialog.create("GECIquant Ver 1.0");
                    Dialog.addMessage("DETECTION OF GENETICALLY ENCODED CALCIUM SIGNALS");
                    Dialog.addMessage("_____");

                    Dialog.addString("Enter pixel calibration i.e., 1 pixel = ? microns","0.230178");
                    Dialog.addMessage("_____");

                    Dialog.addMessage("SELECT MODULE");
                    IMG_PROCESS = newArray(3);
                    IMG_PROCESS[0] = "BACKGROUND SUBTRACTION";
```

```
IMG_PROCESS[1] = "ROI DETECTION";
IMG_PROCESS[2]="ROI ANALYSES";

Dialog.addRadioButtonGroup("", IMG_PROCESS, 3, 1, IMG_PROCESS[1]);
Dialog.addMessage("_____");

Dialog.addMessage("CHOOSE THE TYPE OF ROI ANALYSES");
ROI_ANALYSES = newArray(6);
ROI_ANALYSES[0] = "ROI TIME PROFILES"
ROI_ANALYSES[1] = "CENTROID DISTANCES AND AREAS";
ROI_ANALYSES[2] = "KYMOGRAPHS";
ROI_ANALYSES[3] = "ROI SEGMENTATION";
ROI_ANALYSES[4] = "EXPANDING SIGNALS/WAVES";
ROI_ANALYSES[5] = "NONE";
Dialog.addRadioButtonGroup("", ROI_ANALYSES,6, 1, ROI_ANALYSES[0]);

Dialog.show();
pixel_cal = Dialog.getString();
CHOICE_IMG_PROCESS = Dialog.getRadioButton();
CHOICE_ROI_ANALYSES = Dialog.getRadioButton();

selectWindow(MainStack);
run("Set Scale...", "distance=1 known=pixel_cal pixel=1 unit=um");

if (CHOICE_IMG_PROCESS == IMG_PROCESS[0]) {
        // ------PERFORM IMAGE PREPROCESSING (BACKGROUND SUBTRACTION)----------
        Img_Preprocess();
}
else if (CHOICE_IMG_PROCESS == IMG_PROCESS[1]) {
        // -------- PERFORM ROI DETECTION -------------------------
        ROI_Detection();
}
else if (CHOICE_IMG_PROCESS == IMG_PROCESS [2]) {
        //--------------------PERFORM ROI ANALYSES ------------
        if (CHOICE_ROI_ANALYSES == ROI_ANALYSES[0]) {
                //-------- GENERATE ROI TIME PROFILES -----------
                ROI_Time_Profile();
        }
        else if (CHOICE_ROI_ANALYSES == ROI_ANALYSES[1]) {
                //-------- COMPUTE CENTROID DISTANCES AND ROI AREAS -----
                ROI_Measurements();
        }
        else if (CHOICE_ROI_ANALYSES == ROI_ANALYSES[2]) {
                //-------- DETECT AND ANALYSE EXPANDING SIGNALS -----------
                ROI_Kymographs();
```

```
                        }
                        else if (CHOICE_ROI_ANALYSES == ROI_ANALYSES[3]) {
                                //-------- PERFORM ROI SEGMENTATION -----
                                ROI_Segmentation();
                        }
                        else if (CHOICE_ROI_ANALYSES == ROI_ANALYSES[4]) {
                                //-------- DETECT AND ANALYSE EXPANDING SIGNALS ----------
                                Expanding_Signals();
                        }
                        else
                                exit("You did not make a valid choice");
                }
                exit();


//*******************************************************************************************************
// MODULE-1:            IMAGE PREPROCESSING
// DESCRIPTION:         Performs background subtraction in a user-interactive manner.
// FUNCTIONS:           Img_Preprocess()
//                      Subtract_Background ()
// UPDATED ON:          June 2014
//*******************************************************************************************************

// ----------Preprocesses image t-stack to subtract background
function Img_Preprocess () {
        // Clearing the Results window and Analyze menu – Set Measurements selections
        run("Set Measurements...", "redirect=None decimal=3");
        run("Clear Results");

        selectWindow(MainStack);
        run("Z Project...", "projection=[Average Intensity]");
        STDtitle = getTitle();
        StackTitle = Subtract_Background();
        selectWindow(STDtitle);
        run("Close");
        selectWindow("ROI Manager");
        run("Close");

        return;
}

// --------------Calculates the background pixel value for every frame as the average of pixel intensities
// in the user selected area
function Subtract_Background () {
        selectWindow(STDtitle);
```

```
frame_width = getWidth();
frame_height = getHeight();
makeOval(0, 0, 50, 50);
waitForUser("Draw a circle in a region to be used for background subtraction. Click OK");
roiManager("Add");
roiManager("select",0);
// Get the xy coordinates of the ROI that has been set on the Z-projected image
getSelectionBounds(sel_x, sel_y, sel_width, sel_height);
roiManager("Delete");
// Make another image stack
selectWindow(MainStack);
run("Duplicate...", "title=Background_Subtracted duplicate range=1-Stacklen");
StackTitle = getTitle();
Stacklen = nSlices;

img_pix = newArray(frame_width*frame_height);
new_pix = newArray(frame_width*frame_height);
selectWindow(StackTitle);
for (i=1; i<=Stacklen; i++) {
	// Get the avg intensity in that area
	setSlice(i);
	makeOval(sel_x, sel_y, sel_width, sel_height);
	roiManager("Add");
	roiManager("select",0);
	getSelectionCoordinates(subx, suby);
	roiManager("Delete");
	pixval = newArray(subx.length);
	for (k=0; k<subx.length; k++) {
		// Get the pixel value
		pixval[k] = getPixel(subx[k],suby[k]);
	}

	// Find average of the pixel values
	Array.getStatistics(pixval, pix_min, pix_max, pix_mean, pix_stdDev);

	// Subtract that value from all the pixels of that frame
	for (n=0; n<frame_width; n++) {
		for (m=0; m<frame_height; m++) {
			// Subtract the above calculated average
			img_pix[m] = getPixel(n,m);
			new_pix[m] = img_pix[m]-pix_mean;
			setPixel(n,m, new_pix[m]);
		}
	}
}
```

```
	return StackTitle;
}

//*************************************************************************************************
// MODULE-2:		DETECTION OF SOMATA AND PERIPHERAL ROI (e.g., MICRODOMAINS)
// DESCRIPTION:		Detects spatiotemporal profiles of Ca2+ signals in distinct morphological compartments of one
//			or more astrocytes in a user-interactive manner.
// FUNCTIONS:		ROI_Detection ()
//			Soma_Detection ()
//			Microdomain_Detection()
//			thresh_printlog()
//			closeWindows ()
// UPDATED ON:		June 2014
//*************************************************************************************************

// -------------------- Sets parameters for ROI detection
function ROI_Detection () {
	// Clearing the Results window and Analyze menu – Set Measurements selections
	run("Set Measurements...", "redirect=None decimal=3");
	run("Clear Results");

	// User interface to log in parameters for soma detection and analyses
	Dialog.create("GECIquant Ver 1.0");
	Dialog.addMessage("ROI DETECTION");
	Dialog.addMessage("_____________________________________________________");
	ROI_Type = newArray("Soma ROI", "Microdomain ROI");
	Dialog.addRadioButtonGroup("",ROI_Type, 2, 1, ROI_Type [0]);
	Dialog.addMessage("ROI AREA CRITERIA");
	Dialog.addString("Set Somatic ROI area range (sq. micron)", "30");
	Dialog.addString("", "Infinity");
	Dialog.addString("Enter the ROI label prefix", "");
	Dialog.show();

	CHOICE_ROI_Type = Dialog.getRadioButton();
	Min_area = Dialog.getString();
	Max_area = Dialog.getString();
	prefix = Dialog.getString();

	if (CHOICE_ROI_Type== ROI_Type[0])
		Soma_Detection(prefix);
	else if (CHOICE_ROI_Type== ROI_Type[1])
		Microdomain_Detection(prefix);
	return;
}
```

```
// ---------Soma Detection -----------
function Soma_Detection(prefix) {
	selectWindow(StackTitle);
	run("Z Project...", "projection=[Max Intensity]");
	run("Smooth");
	STDtitle = getTitle();
	setAutoThreshold("Default dark");
	run("Threshold...");
	getThreshold(auto_lower,auto_upper);
	run("ROI Manager...");
	waitForUser("1. Adjust threshold \n 2. Draw polygon(s) around the astrocyte(s) of interest \n 3. Add each
polygon to the ROI Manager \n 4. Click OK");
	cellcnt = roiManager("count");
	polycnt = newArray(cellcnt);
	for (i=0; i<cellcnt; i++)
		polycnt[i] = i;
	getThreshold(user_lower,user_upper);

	for (roi_pos=0; roi_pos<cellcnt; roi_pos++) {
		selectWindow(STDtitle);
		roiManager("select", roi_pos);
		run("Analyze Particles...", "size=Min_area-Max_area circularity=0.00-1.00 show=Outlines add stack");
	}
	roiManager("Select",polycnt);
	roiManager("Delete");
	cellcnt = roiManager("count");
	for (roi_pos=0; roi_pos<cellcnt; roi_pos++) {
		roiManager("select", roi_pos);
		roiManager("Rename",prefix+"_"+(roi_pos+1));
	}
	ROI_Time_Profile();
	thresh_printlog ("Soma");
	closeWindows();

	return;
}

// ----------- Microdomain detection ----------------------------------------------------
function Microdomain_Detection(prefix) {
	selectWindow(StackTitle);
	run("Z Project...", "projection=[Max Intensity]");
	run("Smooth");
	STDtitle = getTitle();
	setAutoThreshold("Default dark");
```

```
	run("Threshold...");
	getThreshold(auto_lower,auto_upper);
	run("ROI Manager...");
	waitForUser("1. Adjust threshold \n 2. Draw polygon(s) around the astrocyte(s) of interest \n 3. Add each
polygon to the ROI Manager \n 4. Click OK");
	cellcnt = roiManager("count");

	if (cellcnt > 0) {
		polycnt = newArray(cellcnt);
		for (i=0; i<cellcnt; i++) {
			polycnt[i] = i;
		}
		getThreshold(user_lower,user_upper);
		roiManager("select", polycnt);
		selectWindow(STDtitle);
		run("Analyze Particles...","size=Min_area-Max_area circularity=0.00-1.00 show=Outlines add stack");

		for (roi_pos=0; roi_pos<cellcnt; roi_pos++) {
			roiManager("select", roi_pos);
			roiManager("Rename","Territory_"+(roi_pos+1));
		}

		roicnt = roiManager("count");
		for (roi_pos=cellcnt; roi_pos<roicnt; roi_pos++) {
			roiManager("select", roi_pos);
			roiManager("Rename",prefix+"_"+(roi_pos));
		}

		ROI_Time_Profile();
		thresh_printlog("Microdomain");
	}
	else
		exit("Please draw a polygon around the astrocyte territory and restart");
	closeWindows();

	return;
}

// ------------ Prints the image threshold applied during ROI detection into the log window ------------------
function thresh_printlog (name) {
	print("\\Clear");
	print(name);
	print("Auto_Lower","User_Lower","Auto_Upper","User_Upper");
	print(auto_lower,user_lower,auto_upper,user_upper);
	return;
```

```
}

// ------------ Closes unwanted windows ---------------------------------------------
function closeWindows () {
	// ---------Closing unwanted windows ---------
	if (isOpen("Summary") == true) {
		selectWindow("Summary");
		run("Close");
	}
	if (isOpen("STD_" + StackTitle) == true) {
		selectWindow("STD_" + StackTitle);
		run("Close");
	}
	if (isOpen("MAX_" + StackTitle) == true) {
		selectWindow("MAX_" + StackTitle);
		run("Close");
	}
	if (isOpen("AVG_" + StackTitle) == true) {
		selectWindow("AVG_" + StackTitle);
		run("Close");
	}
	if (isOpen("Drawing of MAX_" + StackTitle) == true) {
		selectWindow("Drawing of MAX_" + StackTitle);
		run("Close");
	}
	if (isOpen("Threshold") == true) {
		selectWindow("Threshold");
		run("Close");
	}
	if (isOpen("Summary*") == true) {
		selectWindow("Summary*");
		run("Close");
	}
	if (isOpen("MAX_Expanding") == true) {
		selectWindow("MAX_Expanding");
		run("Close");
	}
	// ----------  Reposition stacks and result plots/windows  -------------

	if (isOpen(StackTitle) == true) {
		selectWindow(StackTitle);
		getLocationAndSize(x_pos, y_pos, win_width, win_height);
		x_pos=10; y_pos=10;
		setLocation(x_pos, y_pos);
	}
```

```
	if (isOpen("ROI*") == true) {
		selectWindow("ROI*");
		y_pos=win_width+10;
		setLocation(x_pos, y_pos);
	}
	if (isOpen("Results") == true) {
		selectWindow("Results");
		x_pos=0; y_pos=win_height+1;
		win_width=300; win_height=300;
		setLocation(x_pos, y_pos);
	}

	return;
}

//*************************************************************************************************
// MODULE-3A:		ROI ANALYSES – ROI TIME PROFILES
// DESCRIPTION:		Calculates the ROI time profiles and prints in the Results view
// FUNCTIONS:		ROI_Time_Profile ()
// UPDATED ON:		June 2014
//*************************************************************************************************

//--------------- Calculates the time profiles of ROI in the ROI manager ------------------------
function ROI_Time_Profile () {
	run("Clear Results");
	run("Set Measurements...", "  redirect=None decimal=3");

	selectWindow(StackTitle);
	roi_count = roiManager("count");
	stacklen = nSlices;

	time_base = newArray(stacklen);
	avg = newArray(roi_count*stacklen);
	index = 0;
	for (i=0; i<roi_count; i++) {
		selectWindow(StackTitle);
		roiManager("select",i);
		label = call("ij.plugin.frame.RoiManager.getName", i);
		getSelectionCoordinates(xcoord,ycoord);
		for (z=0; z<stacklen; z++) {
			setZCoordinate(z);
			value = newArray(xcoord.length);
			for (x=0; x<xcoord.length; x++)
				value[x] = getPixel(xcoord[x],ycoord[x]);
			Array.getStatistics(value,val_min, val_max, avg[index],val_std);
```

```
                    time_base[z] = z;

                    setResult("ROI_Index", index, i+1);
                    setResult("Z-Value", index, z+1);
                    setResult("ROI Label", index, label);
                    setResult("Intensity", index, avg[index]);

                    index++;
                }
                updateResults;
        }

        return;
}

//*************************************************************************************************
// MODULE-3B:       ROI ANALYSES – CENTROID DISTANCES AND AREAS
// DESCRIPTION:     Computes and prints: i) the distances between the centroids of a reference ROI and the other
//                  ROI, ii) ROI areas.
// FUNCTIONS:       ROI_Measurements ()
//                  Calc_Centroid_Distances ()
//                  Calc_Area ()
// UPDATED ON:      June 2014
//*************************************************************************************************

// ----------------- Generates the user interface to log in parameters ---------------------
function ROI_Measurements () {
        Dialog.create("GECIquant Ver 1.0");
        Dialog.addMessage("CENTROID DISTANCES FROM SOMA AND ROI AREAS");
        Dialog.addMessage("_____________________________________________________________________________");

        Dialog.addString("Enter reference ROI label","SOMA");
        Dialog.show();
        ref_ROI = Dialog.getString();;

        dist_to_ref  = newArray(roi_cnt);
        dist_to_ref = Calc_Centroid_Distances(ref_ROI);
        roi_area = Calc_Area();
        // Tabulate the ROI name and the corresponding distance in the Results window
        run("Clear Results");
        for (i=0; i<roi_cnt; i++) {
                setResult("ROI",i,roi_label[i]);
                setResult("DISTANCE (micron)",i,dist_to_ref[i]);
                setResult("ROI AREA (sq. micron)",i,roi_area[i]);
                updateResults;
```

```
	}

	return;
}

// ----------Computes distances between ROI centroids ----------
function Calc_Centroid_Distances (ref_label) {
	// For each ROI get the centroid into the results window
	selectWindow(StackTitle);
	run("Set Measurements...", "  redirect=None decimal=3");
	run("Set Measurements...", "  centroid redirect=None decimal=3");

	roi_cnt = roiManager("count");
	if (roi_cnt<=2)
		showMessage("There are not enough ROIs to measure distance!");
	else if (roi_cnt>2) {
		ref_index =0;
		roi_list = newArray(roi_cnt);
		roi_label = newArray(roi_cnt);
		for (i=0; i<roi_cnt; i++) {
			roiManager("select", i);
			roi_list[i] = i;
			roi_label[i] = call("ij.plugin.frame.RoiManager.getName", i);
			if (roi_label[i] == ref_label)
				ref_index = i;
		}
	}

	roiManager("Select", roi_list);
	roiManager("Measure");

	// Read each row of the results window and obtain the distance between the largest ROI and the rest
	ref_centroid_x = getResult("X",ref_index);
	ref_centroid_y = getResult("Y",ref_index);
	dist_to_ref = newArray(roi_cnt);
	Xc = newArray(roi_cnt);
	Yc = newArray(roi_cnt);

	// Measure the distance from soma ROI center to microdomain centers
	for (i=0; i<roi_cnt; i++) {
		Xc[i] = getResult("X",i);
		Yc[i] = getResult("Y",i);
		dist_to_ref[i] = sqrt(((ref_centroid_x-Xc[i])*(ref_centroid_x-Xc[i]))+((ref_centroid_y-
Yc[i])*(ref_centroid_y-Yc[i])));
	}
```

```
	return dist_to_ref;
}

// ------------Calculates ROI areas
function Calc_Area () {
	if (isOpen("ROI Manager")) {
		//run("Set Scale...");
		n = roiManager("count");
		roi_area = newArray(n);
		for (i=0; i<n; i++) {
			roiManager("select", i);
			getStatistics(roi_area[i]);
		}
	}
	else
		showMessage("Open the ROI Manager and then run the macro");
	return roi_area;
}

//*************************************************************************************************
// MODULE-3C:		ROI ANALYSES – KYMOGRAPHS
// DESCRIPTION:		Generates Kymographs of ROI intensity changes in time. ROI can be either arranged according to
//			their X-Y coordinates or in the order of increasing distance from a reference ROI such as Soma
//			along the Y-axis of the Kymograph.
// FUNCTIONS:		ROI_Kymographs ()
//			Kym_XYcoord ()
//			Kym_dist_order ()
// UPDATED ON:		June 2014
//*************************************************************************************************

// ------------Generates the user interface for Kymograph creation----------------------------------------------
function ROI_Kymographs () {
	Dialog.create("GECIquant Ver 1.0");
	Dialog.addMessage("KYMOGRAPHS");
	Dialog.addMessage("_____________________________________________________________________________");

	GRAPH = newArray(2);
	GRAPH[0]="MULTIPLE ROI: PIXEL ROWS SORTED BASED ON CENTROID DISTANCES FROM REFERENCE ROI";
	GRAPH[1]="MULTIPLE ROI: XY-COORD ROI INTENSITIES";
	Dialog.addRadioButtonGroup("Select an activity to profile\n", GRAPH, 2, 1, GRAPH[0]);
	Dialog.addString("Enter stack or sub-stack size:", 300);
	Dialog.addString("Enter reference ROI label:", "SOMA");

	Dialog.show();
```

```
	CHOICE = Dialog.getRadioButton();
	stacklen = Dialog.getString();
	ref_label = Dialog.getString();

	if (isOpen("ROI Manager")) {
		roi_count = roiManager("count");
		if (roi_count < 10)
			showMessage("At least 10 ROI are needed to make the kymographs");
		else {
			orig_dist = newArray(roi_count);
			sorted_dist = newArray(roi_count);
			roi_index = newArray(roi_count);
			roi_avg= newArray(stacklen);

			if (CHOICE == GRAPH[0]) {
				orig_dist = Calc_Centroid_Distances (ref_label);
				sorted_dist = Array.copy(orig_dist);
				Array.sort(sorted_dist);
				roi_index = Array.rankPositions(orig_dist);
				Kym_dist_order(roi_count);
			}
			else if (CHOICE == GRAPH[1])
				Kym_XYcoord();
			else
				exit();
		}
	}
	else
		showMessage("Open ROI manager, load the ROI to be used and restart");

	return;
}

// ---------- Creates Kymograph of ROI intensities with ROI plotted according to their X-Y coordinates along the Y-axis
function Kym_XYcoord() {
	setBatchMode(true);
	newImage("KYMOGRAPH BASED ON XY COORDINATES", "8-bit", stacklen, roi_count*4, 1);
	graph_title = getTitle();
	k=0;
	for (i=0; i<roi_count; i++) {
		selectWindow(MainStack);
		roiManager("select",i);
		getSelectionCoordinates(xcoord,ycoord);
		m=0;
```

```
		for (z=0; z<stacklen; z++) {
			selectWindow(MainStack);
			setZCoordinate(z);
			value = newArray(xcoord.length);
			for (j=0; j<xcoord.length; j++)
				value[j] = getPixel(xcoord[j],ycoord[j]);
			Array.getStatistics(value,val_min, val_max, avg,val_std);
			selectWindow(graph_title);
			setPixel(z, k, val_max);
			setPixel(z, k+1, val_max);
			setPixel(z, k+2, val_max);
			setPixel(z, k+3, val_max);
		}
		k=k+4;
	}

	setBatchMode(false);

	return;
}

// ------Creates a Kymograph of ROI intensities with ROI plotted in increasing order of distance from a reference ROI such
// as soma along the Y-axis----------
function Kym_dist_order(roi_count) {
	setBatchMode(true);
	roi_cnt = roi_count;
	roi_avg= newArray(stacklen);
	roi_profile=newArray(stacklen);

	newImage("KYMOGRAPH BASED ON SORTED CENTROID DISTANCES FROM REFERENCE", "8-bit", stacklen,
roi_cnt*4, 1);
	kymograph = getTitle();
	k=0;
	for (i=0; i<lengthOf(sorted_dist); i++) {    // For each ROI sorted based on distance from ref ROI
		selectWindow(MainStack);
		roiManager("select",roi_index[i]);          // Select the ROIs based on distance from ref ROI
		getSelectionCoordinates(xcoord,ycoord);           // Get the ROI's X_Y corordinates
		pix_value = newArray(xcoord.length);    // Create a new array to hold the pixel intensities of the
selected ROI in each frame of the stack
		m=0;
		for (z=0; z<stacklen; z++) {                    // In each frame os the stack, do the following
			selectWindow(MainStack);
			setZCoordinate(z);
			for (j=0; j<xcoord.length; j++)
```

```
                                pix_value[j] = getPixel(xcoord[j],ycoord[j]);        // Fill the pix_value array at that stack
pos within the selected ROI
                        Array.getStatistics(pix_value,val_min, val_max, avg,val_std);
                        selectWindow(kymograph);
                        //for (x=m; x<m+4; x++) {
                                for (y=k; y<k+4; y++) {
                                        setPixel(z, y, val_max);
                                }
                        //}
                        //m=m+4;
                }
                k=k+4;

        }
        setBatchMode(false);

        return;
}

//*************************************************************************************************
// MODULE-3D:           ROI ANALYSES – ROI SEGMENTATION
// DESCRIPTION:         Dissects the ROI into sub-ROI based on 1) spatial, and 2) spatio-temporal approaches.
// FUNCTIONS:           ROI_Segmentation ()
//                      spatiotemporal_SubROI()
//                      Rectangular_Segments()
//                      drawROIs ()
//                      seg_kymograph ()
//                      area_printlog()
// UPDATED ON:          June 2014
//*************************************************************************************************

// --------------Segmentation of ROI -------------------------------------------------
function ROI_Segmentation () {
        // Clearing the Results window and Analyze menu – Set Measurements selections
        run("Set Measurements...", "redirect=None decimal=3");
        run("Clear Results");

        // User interface to log in parameters for soma detection and analyses
        Dialog.create("GECIquant Ver 1.0");
        Dialog.addMessage("SPATIOTEMPORAL SEGMENTS OF A LARGER ROI (e.g., SOMA)");
        labels = newArray("Rectangular Segmentation", "Sub-ROIs","None");
        Dialog.addRadioButtonGroup("", labels, 3, 1, labels[2]);
        Dialog.addString("Height of the rectangle (micron)", "20");
        Dialog.addString("Set minimum sub-ROI area (sq. micron)", "10");
        Dialog.addString("Set fraction of soma area for sub-ROI deletion", "0.6");
```

```
	Dialog.show();

	rect_ht = Dialog.getString();
	min_area = Dialog.getString();
	persubroi_delete = Dialog.getString();

	SomaSegmentation = Dialog.getCheckbox();
	seg_method = Dialog.getRadioButton();
	//-------------------------------------------------------------------------------------------------------------------

	if (seg_method==labels[1]) {
		// ----------------SUB-ROI SEGMENTS ------------
		spatiotemporal_SubROI();
	}
	else if (seg_method==labels[0]) {
	// ---------RECTANGULAR SEGMENTS-------------
		cellcnt = roiManager("count");
		if (cellcnt>1) {
			for (i=0; i<cellcnt-1; i++) {
				Rectangular_Segments(i);
			}
		}
		else if (cellcnt == 1)
			Rectangular_Segments(0);
	}

	return;
}

// ------------------ Function to detect sub-ROI of soma based on spatio-temporal characteristics --------------------------------
// This function provides a means to segment the somatic signal into spatially distinct subset of ROI that are extracted
// based on the temporal characteristics. Every 3 frames of the stack is averaged. In the reduced sub-stack, possible sub-
// ROI  within the main somatic ROI are detected for every frame. In the resulting ROI set, first all the sub-ROI that
// encompass > user-set percent area of the main somatic ROI are deleted. Next, spatially non-overlapping ROI, are
// extracted by comparing the x-y coordinates of each ROI with the other ROI and deteling all the ROI that spatially
// overlap.

function spatiotemporal_SubROI() {
	// ---------- Generate sub ROIs by averaging 3 frames ----------------
	selectWindow(StackTitle);
	run("Grouped Z Project...", "projection=[Standard Deviation] group=3");
	subStackTitle = getTitle();
	cnt = roiManager("count");
	if (cnt == 1) {
		selectWindow(subStackTitle);
```

```
roiManager("select", 0);		// Select the somatic ROI (or S-ROI)
getStatistics(soma_area, soma_mean);	// Get the area covered by S-ROI
setAutoThreshold("Default dark");		// Threshold the substack
run("Analyze Particles...", "size=min_area-soma_area circularity=0.00-1.00 show=Outlines display add
stack");
roi_count = roiManager("count");
num = 0;
roi_del = newArray(roi_count);

// ------------------------ Set deletion flags for repeating ROI --------------------------
num=0;
for (i=1;  i<roi_count-1; i++) {		// For each ROI
	roiManager("select",i);
	getStatistics(roi_area,roi_mean);
	if (roi_area > (persubroi_delete)*(soma_area)) {
		roi_del[num]=i;
		num++;
	}
}
if (num > 0) {
	delROI = newArray(num);
	for (k=0; k<num; k++)
		delROI[k] = roi_del[k];
	roiManager("select", delROI);
	roiManager("Delete");
}

roi_count = roiManager("count");
num=0;
for (i=1;  i<roi_count-1; i++) {		// For each ROI
	roiManager("select",i);
	getSelectionCoordinates(xval,yval);
	getStatistics(roi_area,roi_mean);
	for (j=i+1; j<roi_count; j++) {
		roiManager("select",j);
		distinct = 0;
		for (k=0; k<xval.length; k++) {
			region = selectionContains(xval[k], yval[k]);
			if (region == 1)
				distinct++;
		}
		if (distinct > 0) {
			roi_del[num]=j;
			num++;
		}
```

```
                }
                if (num > 0) {
                        delROI = newArray(num);
                        for (k=0; k<num; k++)
                                delROI[k] = roi_del[k];
                        roiManager("select", delROI);
                        roiManager("Delete");
                }
                roi_count = roiManager("count");
                num=0;
        }

        roi_cnt = roiManager("count");
        for (i=1; i<roi_cnt; i++) {
                roiManager("select",i);
                roiManager("Rename","Seg_"+i);
        }

        drawROIs("SUB-ROI SEGMENTS");
        ROI_Time_Profile();
        seg_kymograph();
        area_printlog();
        closeWindows();
    } // (cnt == 1)

    return;
}

// ---------- Function to create spatial segments of the somatic ROI ----------------
// A bounding rectangle is placed around the soma ROI. Soma is segmented by creating smaller rectangles of equal
// heights pre-specified by the user.  Temporal profiles of segmented sub-ROI of soma are generated. ROI intensity
// kymograph is generated.

function Rectangular_Segments(roi_pos) {
    selectWindow(StackTitle);
    rect_ht_pix = (rect_ht)/(pixel_cal);        // Converting micron to pixels
    roiManager("select", roi_pos);
    getSelectionBounds(roi_x, roi_y, roi_width, roi_height);
    getSelectionCoordinates(roi_xCoordinates, roi_yCoordinates);
    ymax = getHeight();
    xmax = getWidth();

    run("Z Project...", "projection=[Max Intensity]");
    run("Smooth");
    MAXtitle = getTitle();
```

```
	selectWindow(MAXtitle);
	setAutoThreshold("Default dark");
	run("Threshold...");
	waitForUser("Adjust threshold values as set for the detection of the large ROI to be segmented\n Click OK");
	getThreshold(lower,upper);
	print("Lower threshold:", lower);
	print("Upper threshold:", upper);

	ht = roi_y;
	for (ht = roi_y; ht < (roi_y + roi_height); ht = ht+rect_ht_pix) {
		selectWindow(MAXtitle);
		roiManager("select", 0);
		makeRectangle(roi_x, ht, roi_width, rect_ht_pix);
		run("Analyze Particles...", "size=1-soma_area circularity=0.00-1.00 show=Outlines add stack");
		close("Draw*");
		close("Summ*");
		close("Res*");
		selectWindow(MAXtitle);
		roiManager("Add");
	}
	roi_count = roiManager("count");
	num = 0;
	roi_del = newArray(roi_count);
	// ----------Set deletion flags for the rectangular ROIs and for segmented ROIs that are not within the ROI to be
segmented---
	for (i=0;i<roi_count; i++) {
		roiManager("select",i);
		getSelectionBounds(x, y, roi_width, roi_height);
		getSelectionCoordinates(xval,yval);
		if ((roi_width == roi_width) && (roi_height == rect_ht_pix)) {
			roi_del[num]=i;
			num++;
		}
		else {
			flag=0;
			for (k=0; k<xval.length; k++) {
				roiManager("select",0);
				region = selectionContains(xval[k], yval[k]);
				if (region == 1) {
					flag++;
				}
			}
			if (flag == 0) {
				roi_del[num]=i;
				num++;
```

```
                        }
                }
        }
        if (num > 0) {
                delROI = newArray(num);
                for (k=0; k<num; k++)
                        delROI[k] = roi_del[k];
                roiManager("select", delROI);
                roiManager("Delete");
        }

        roi_cnt = roiManager("count");
        for (i=1; i<roi_cnt; i++) {
                roiManager("select",i);
                roiManager("Rename","Seg_"+i);
        }

        drawROIs("RECTANGULAR SEGMENTS");
        ROI_Time_Profile();
        seg_kymograph();
        area_printlog();
        closeWindows();

        return;
}

// ---------------- Overlays the sub-ROIs of soma -------------------------
function drawROIs (title) {
        call("ij.gui.ImageWindow.setNextLocation", 0, 0);
        newImage(title, "16-bit", width, height, 1);
        selectWindow(title);
        run("Set Scale...", "distance=1 known=pixel_cal pixel=1 unit=um");
        setColor(0,0,0);
        fill();
        color = "red";
        for (i=0; i<roiManager("count"); i++) {
                roiManager("select",i);
                selectWindow(title);
                Overlay.addSelection("",0,color);
                if (color == "red")
                        color = "blue";
                else if (color == "blue")
                        color = "green";
                else if (color == "green")
                        color = "orange";
```

```
		else if (color == "orange")
			color = "magenta";
		else if (color == "magenta")
			color = "yellow";
		else if (color == "yellow")
			color = "red";
	}

	return;
}

// ------------Prints the sub-ROI areas into the log window ---------------------------------
function area_printlog () {
	print("ROI Label", "Area1 (sq. micron)", "Area2 (pixel units)");
	for (i=0; i<roiManager("count"); i++) {
		selectWindow(StackTitle);
		roiManager("select", i);
		roi_label = call("ij.plugin.frame.RoiManager.getName", i);
		getStatistics(area);
		getRawStatistics(rawarea);
		print(roi_label, area, rawarea);
	}

	return;
}

// ---- Function to generate ROI intensity kymograph of all the ROI in the ROI manager -----
function seg_kymograph () {
	setBatchMode(true);
	selectWindow(StackTitle);
	stacklen = nSlices;

	roi_count = roiManager("count");
	roi_avg= newArray(stacklen);

	newImage("KYMOGRAPH OF ROI SEGMENTS", "8-bit", stacklen*4, roi_count*6, 1);
	k=0;
	for (i=0; i<roi_count; i++) {
		selectWindow(StackTitle);
		roiManager("select",i);
		getSelectionCoordinates(xcoord,ycoord);
		m=0;
		for (z=0; z<stacklen; z++) {
			selectWindow(StackTitle);
			setZCoordinate(z);
```

```
                        value = newArray(xcoord.length);
                        for (j=0; j<xcoord.length; j++)
                                value[j] = getPixel(xcoord[j],ycoord[j]);
                        Array.getStatistics(value,val_min, val_max, avg,val_std);
                        selectWindow("KYMOGRAPH OF ROI SEGMENTS");
                        for (x=m; x<m+4; x++) {
                                for (y=k; y<k+6; y++) {
                                        setPixel(x, y, val_max);
                                }
                        }
                m=m+4;
                }
                k=k+6;
        }
        setBatchMode(false);
}

//*************************************************************************************************
// MODULE-3E:           ROI ANALYSES – EXPANDING SIGNALS/WAVES
// DESCRIPTION:         Allows creation of sub-stacks to isolate wave-like signals and re-detect the expanding and
//                      contracting ROI within a larger ROI. Prints the time and area profiles of such signals.
// FUNCTIONS:           Expanding_Signals ()
//                      detect_ESpattern ()
//                      ES_writeResults ()
// UPDATED ON:          June 2014
//*************************************************************************************************

/* NOTE: The Expanding Signal (ES) ROI that marks the contour detected a priori. The user is asked to enter the label
of this ROI. Within this ROI, and for the substack consisting of the ES frames only, each frame of the stack is thresholded
and the user is allowed to adjust the threshold. Once the threshold adjustment is done, the sub-ROI of the ES are
detected with the new area criteria. The ROI areas, mean intensity and integrated densities are printed in the Results
window. */

// ----------User interface for isolating waves ------------------------------
function Expanding_Signals () {
        // Clearing the Results file
        run("Set Measurements...", "redirect=None decimal=3");
        run("Clear Results");

        Dialog.create("GECIquant Ver 1.0");
        Dialog.addMessage("EXPANDING SIGNALS");
        Dialog.addMessage("_____________________________________________________________________________");
        ES_ANAL = newArray("DETECT EXPANDING SIGNAL PATTERN", "PRINT EXPANDING SIGNAL MEASUREMENTS");
        Dialog.addRadioButtonGroup("", ES_ANAL, 2, 1, ES_ANAL[0]);
        Dialog.addMessage("_____________________________________________________________________________");
```

```
	Dialog.addMessage("Enter the following to detect the pattern of Expanding Signal");
	Dialog.addString("Enter the ROI label for the Expanding Signal", "ES-1");
	Dialog.addString("Enter START frame", "1");
	Dialog.addString("Enter STOP frame", "10");
	Dialog.addString("ROI area range (sq. microns)","2");
	Dialog.addString("","Infinity");
	Dialog.addString("Enter prefix for expanding ROI","ES");

	Dialog.show();

	CHOICE = Dialog.getRadioButton();
	ES_LABEL = Dialog.getString();
	START_FRAME = Dialog.getString();
	STOP_FRAME = Dialog.getString();
	AREA1 = Dialog.getString();
	AREA2 = Dialog.getString();
	ES_prefix = Dialog.getString();

	// Check if the ROI Manager is open.
	if (isOpen("ROI Manager")) {
		selectWindow("ROI Manager");
		if (roiManager("count") == 0) {
			roiManager("Open", "");
		}
	}
	else {
		roiManager("Open", "");
	}

	if (CHOICE == "DETECT EXPANDING SIGNAL PATTERN") {
		detect_ESpattern(ES_prefix);
	}
	else if (CHOICE == "PRINT EXPANDING SIGNAL MEASUREMENTS") {
		substack = StackTitle;
		ES_writeResults();
	}

	closeWindows();
	return;
}

// ---------------- Detects the sub-ROI of the larger Wave-encompassing ROI
function detect_ESpattern(ES_prefix) {
	// Create a substack consisting of the Expanding Signal Pattern
	selectWindow(StackTitle);
```

```
run("Duplicate...", "title=Expanding Signal duplicate range=START_FRAME-STOP_FRAME");
substack = getTitle();

// Check whether the labeled ROI is in the ROI Manager
ROI_FOUND = 0; ES_INDEX = 0;
for (i=0; i<roiManager("count"); i++) {
	roiManager("select", i);
	roi_label = Roi.getName();
	if (roi_label == ES_LABEL) {
		ROI_FOUND = 1;
		ES_INDEX = i;
	}
}

if (ROI_FOUND == 1) {
	selectWindow(substack);
	run("Z Project...", "projection=[Max Intensity]");
	run("Smooth");
	MAXtitle = getTitle();
	roiManager("select",ES_INDEX);
	selectWindow(MAXtitle);
	run("Threshold...");
	waitForUser("Adjust threshold and Click OK");
	getThreshold(lower, upper);
	print("Lower threshold:" + lower + "\n");
	print("Upper threshold:" + upper + "\n");
	selectWindow(substack);
	substack_size = nSlices;

	// Detect possible ROIs in each frame

	selectWindow(substack);
	run("Threshold...");
	setThreshold(lower, upper);
	roiManager("select",ES_INDEX);
	selectWindow(substack);
	run("Analyze Particles...", "size=AREA1-AREA2 circularity=0.00-1.00 show=Outlines add stack");

	roi_cnt = roiManager("count");
	for (i=1; i<roi_cnt; i++) {
		roiManager("select",i);
		z = getSliceNumber();
		roiManager("Rename",ES_prefix+"_"+i+"_Slice_"+z);
	}
```

```
		ES_writeResults();
	}
	else {
		showMessage("Could not find " + ES_LABEL + "or the roiManager is empty" + \n + "Please restart");
		exit();
	}
}

// ---------------- Measures and prints the area profile of an expanding wave-like signal
function ES_writeResults () {
	run("Set Measurements...", "area mean integrated stack redirect=None decimal=2");
	roi_cnt = roiManager("count");
	ES_roi = newArray(roi_cnt);
	ES_label = newArray(roi_cnt);
	ES_property = newArray(roi_cnt);
	slice_num=1; prev_num=1; area=newArray(roi_cnt); mean=newArray(roi_cnt);
	tot_area = 0; mean_AU = 0; k=0;
	selectWindow(substack);
	for (i=0; i<roi_cnt; i++) {
		roiManager("select",i);
		ES_label[i] = call("ij.plugin.frame.RoiManager.getName", i);
		getStatistics(area[i],mean[i]);
		setResult("ROI Label", i, ES_label[i]);
		setResult("Area (um2)", i, area[i]);
		setResult("Mean Intensity (AU)", i, mean[i]);
		setResult("Integrated Density (um2 - AU)", i, area[i]*mean[i]);
		setResult("Frame #", i, getSliceNumber());
	}
	updateResults;

	return;
}
```